DETAILED ACTION

Response to Amendment

1. This communication is in response to the amendment filed on August 25, 2008.

Claims 1, **9-10**, **12**, **20-21**, and **26-27** have been amended.

Claims 29-38 have been added.

Claims 1-10, **12-21**, and **25-38** are pending in this instant Office action.

Response to Arguments

2. **First**, Applicant's argument towards **claims 1**, **12**, and **26-27** regarding the fact that Rawat does not teach or disclose "*dynamically adding the basic data type of the variable with metadata at runtime*".

The Examiner respectfully disagrees with the above remark. Accordingly, Rawat teaches filling out electronic forms automatically, in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data, without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention, [Column 4, Lines 56-62]). This citation clearly anticipates the argued limitation above.

3. **Second**, Applicant's argument towards **claims 1**, **12**, and **26-27** regarding the fact that Rawat does not teach or disclose "*the application program automatically processes changes in the metadata without incurring changes in a computer system*".

The Examiner concurs with the above remark. However, it is noted that Harper et al. (*Pat. No. US 7,093,261, filed on July 25, 2001; hereinafter Harper*) teaches

automatically processes changes in the metadata without incurring changes in a computer system (*Addition of messages or deletion of existing messages is controlled by the database 110, and the applications 102 which use the messages. Neither the MOM modules 104, nor the database 110, needs to be recompiled to add data entities in the database 110 such as Hosts, Applications, Queues, Queue Managers, etc. Messages and MetaData can also be added, modified, or deleted without redesign or recompilation of a system in accordance with one embodiment of the present invention. Further, in one embodiment, these changes can be made while the system is in use, without affecting the operation of other, unchanged parts of the system*, [Column 7, Lines 16-27]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Harper with the teachings of Rawat for the purpose of integrating multiple applications, where the various applications do not need to be aware of the existence of the other applications ([Column 2, Lines 33-36] of Harper).

In view of the above, the Examiner contends that all limitations as recited in the claims have been addressed in this instant Office action. Hence, Applicant's arguments do not distinguish over the claimed invention over the prior art of record.

For the above reasons, the Examiner believed that rejections of this instant Office action is proper.

Claim Objections

4. **Claim 12** is objected to because of the following informalities: grammatical error on line 7. Applicant is suggested to change from "*dynamically adding*" to "*dynamically add*".

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. **Claims 1-3**, **5-8**, **12-14**, **16-19**, and **25-28**, are rejected under 35 U.S.C. 103(a) as being unpatentable over Rawat et al. (*Pat. No. 6,662,340, filed on May 30, 2002; hereinafter Rawat*) in view of Harper et al. (*Pat. No. US 7,093,261, filed on July 25, 2001; hereinafter Harper*).

Regarding **claim 1**, Rawat clearly shows and discloses a computer-implemented method for dynamic data type enrichment (*Abstract*) comprising the steps:

loading an application program in to memory (*Client-side program code examines electronic documents such as web pages and automatically fills in fields of forms contained in the document with the appropriate data from a user profile, without requiring prior mapping or examination of the form*, [Column 3, Lines 61-65]. *It is well inherent that any computer program is loaded into computer's memory in advance to execution*), the application program comprising a variable that is defined as an instance

of both a basic data type and a specific data type (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*);

accessing metadata at runtime to map the variable to a definition of the specific data type (*Figure 4 shows matching the values with the dictionaries previously described to map the fields to metadata, wherein metadata comprises a data type, such as Last name; First name; AddressLine 1; AddressLine 2; and City*, [Column 6, Lines 3-11]);

dynamically adding the basic data type of the variable with metadata at runtime (*filling out electronic forms automatically, in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data ,without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention*, [Column 4, Lines 56-62]); and

processing the variable consistently with the metadata definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301. Also incorporated in the field label dictionary are the rules for mapping the field to the correct metadata and for mapping fields lacking labels to the correct metadata based on the field's context. At the same*

time, the functional blocks of the form are identified, for example, shipping address and billing address. In this way, identical field labels, such as "address" or "zip code" in a shipping address block and a billing address block are mapped to the correct metadata, [Colum 5, Lines 18-32]).

Rawat does not explicitly disclose the application program automatically processes changes in the metadata without incurring changes in a computer system implementing the method.

However, Harper discloses automatically processes changes in the metadata without incurring changes in a computer system (*Addition of messages or deletion of existing messages is controlled by the database 110, and the applications 102 which use the messages. Neither the MOM modules 104, nor the database 110, needs to be recompiled to add data entities in the database 110 such as Hosts, Applications, Queues, Queue Managers, etc. Messages and MetaData can also be added, modified, or deleted without redesign or recompilation of a system in accordance with one embodiment of the present invention. Further, in one embodiment, these changes can be made while the system is in use, without affecting the operation of other, unchanged parts of the system*, [Column 7, Lines 16-27]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Harper with the teachings of Rawat for the purpose of integrating multiple applications, where the various applications do not need

to be aware of the existence of the other applications ([Column 2, Lines 33-36] of Harper).

Regarding **claim 2**, Rawat further discloses the application program uses an application programming interface for accessing the metadata (*locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304*, [Column 5, Lines 18-22]).

Regarding **claim 3**, Rawat further discloses the application program calls through the application programming interface at least one metadata service that relates to the basic data type (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 5**, Rawat further discloses the basic data type is defined in one or more of Visual Basic, Java, or C++ (*other commonly known scripting and programming languages would also be suitable for the invention such as VBSCRIPT, PERL, JAVA, or JPYTHON*, [Column 5, Lines 58-61]).

Regarding **claim 6**, Rawat further discloses the metadata defines an allowed value range for the specific data type that is a subset of an allowed value range for the basic data type (*Figure 1 shows the specific data type of credit card number requires 16 digits, each digit is from the range 0 to 9 defined by the basic data type, i.e. integers that makes up the credit card sequence*).

Regarding **claim 7**, Rawat further discloses the metadata defines a text label for a user input field (*if a field lacks a label, the algorithm may analyze the field's programmatic name. Following field name analysis, the field name is compared to the entries in the Field label dictionary and a match found. The field is then mapped as described above*, [Column 7, Lines 19-23]).

Regarding **claim 8**, Rawat further discloses the variable is set to a value entered into the user input field (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 12**, Rawat clearly shows and discloses a computer system ([Column 5, Lines 3-5]) comprising:

a memory (*laptop or desktop comprises memory*, [Column 5, Lines 3-5]) storing an application program that comprises a variable that is defined as an instance of both a basic data type and a specific data type (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*); and

a processor executing instructions (*laptop or desktops comprises CPU to execute instructions loaded on memory*, [Column 5, Lines 3-5]) to:

access metadata at runtime to map the variable to a definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 18-22]); and

dynamically adding the basic data type of the variable with metadata at runtime (*filling out electronic forms automatically, in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data ,without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention*, [Column 4, Lines 56-62]); and

process the variable consistently with the metadata definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301. Also incorporated in the field label dictionary are the rules for mapping the field to the correct metadata and for mapping fields lacking labels to the correct metadata based on the field's context. At the same time, the functional blocks of the form are identified, for example, shipping address and billing address. In this way, identical field labels, such as "address" or "zip code" in a shipping address block and a billing address block are mapped to the correct metadata*, [Colum 5, Lines 18-32]).

Rawat does not explicitly disclose the application program automatically processes changes in the metadata without incurring changes in a computer system implementing the method.

However, Harper discloses automatically processes changes in the metadata without incurring changes in a computer system (*Addition of messages or deletion of existing messages is controlled by the database 110, and the applications 102 which use the messages. Neither the MOM modules 104, nor the database 110, needs to be recompiled to add data entities in the database 110 such as Hosts, Applications, Queues, Queue Managers, etc. Messages and MetaData can also be added, modified, or deleted without redesign or recompilation of a system in accordance with one embodiment of the present invention. Further, in one embodiment, these changes can be made while the system is in use, without affecting the operation of other, unchanged parts of the system*, [Column 7, Lines 16-27]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Harper with the teachings of Rawat for the purpose of integrating multiple applications, where the various applications do not need to be aware of the existence of the other applications ([Column 2, Lines 33-36] of Harper).

Regarding **claim 13**, Rawat further discloses an application programming interface to access the metadata from the application program (*locating the field labels,*

associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304, [Column 5, Lines 18-22]).

Regarding **claim 14**, Rawat further discloses the application programming interface provides at least one metadata service that relates to the basic data type used by the application program (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 16**, Rawat further discloses the basic data type is defined in one or more of Visual Basic, Java, or C++ (*other commonly known scripting and programming languages would also be suitable for the invention such as VBSCRIPT, PERL, JAVA, or JPYTHON*, [Column 5, Lines 58-61]).

Regarding **claim 17**, Rawat further discloses the metadata defines an allowed value range for the specific data type that is a subset of an allowed value range for the basic data type (*Figure 1 shows the specific data type of credit card number requires 16 digits, each digit is from the range 0 to 9 defined by the basic data type, i.e. integers that makes up the credit card sequence*).

Regarding **claim 18**, Rawat further discloses the metadata defines a text label for a user input field (*if a field lacks a label, the algorithm may analyze the field's programmatic name. Following field name analysis, the field name is compared to the*

entries in the Field label dictionary and a match found. The field is then mapped as described above, [Column 7, Lines 19-23]).

Regarding **claim 19**, Rawat further discloses the variable is set to a value entered into the user input field (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50]).

Regarding **claim 25**, Rawat clearly shows and discloses the variable is declared as an instance of both the basic data type and the specific data type at design time (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*).

Regarding **claim 26**, Rawat clearly shows and discloses a computer-implemented method for dynamic data type enrichment (*Abstract*) comprising:

loading an application program into memory (*Client-side program code examines electronic documents such as web pages and automatically fills in fields of forms contained in the document with the appropriate data from a user profile, without requiring prior mapping or examination of the form*, [Column 3, Lines 61-65]. *It is well inherent that any computer program is loaded into computer's memory in advance to execution*), the application program comprising a variable that is defined as an instance

of both a basic data type and a specific data type (*Figure 1 shows the 'Credit Card Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number*);

accessing metadata over a network at runtime using an application programming interface (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 18-22]), the metadata mapping the variable to a definition of the specific data type that indicates a text label for an input field to the variable (*if a field lacks a label, the algorithm may analyze the field's programmatic name. Following field name analysis, the field name is compared to the entries in the Field label dictionary and a match found. The field is then mapped as described above*, [Column 7, Lines 19-23]) and indicating allowed value range for the variable (*after the visible elements of the form have been completely mapped the correct user data is retrieved from a stored user profile 302, a data file stored on the client, and concatenated, truncated or re-formatted as required by the display format, and the form fields are populated with the data*, [Column 5, Lines 44-50])); and

dynamically adding the basic data type of the variable with metadata at runtime (*filling out electronic forms automatically, in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated*

with the correct user data ,without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention, [Column 4, Lines 56-62]); and

processing the variable consistently with the indicated allowed value range (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301. Also incorporated in the field label dictionary are the rules for mapping the field to the correct metadata and for mapping fields lacking labels to the correct metadata based on the field's context. At the same time, the functional blocks of the form are identified, for example, shipping address and billing address. In this way, identical field labels, such as "address" or "zip code" in a shipping address block and a billing address block are mapped to the correct metadata*, [Colum 5, Lines 18-32]).

Rawat does not explicitly disclose the application program automatically processes changes in the metadata without incurring changes in a computer system implementing the method.

However, Harper discloses automatically processes changes in the metadata without incurring changes in a computer system (*Addition of messages or deletion of existing messages is controlled by the database 110, and the applications 102 which use the messages. Neither the MOM modules 104, nor the database 110, needs to be recompiled to add data entities in the database 110 such as Hosts, Applications,*

Queues, Queue Managers, etc. Messages and MetaData can also be added, modified, or deleted without redesign or recompilation of a system in accordance with one embodiment of the present invention. Further, in one embodiment, these changes can be made while the system is in use, without affecting the operation of other, unchanged parts of the system, [Column 7, Lines 16-27]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Harper with the teachings of Rawat for the purpose of integrating multiple applications, where the various applications do not need to be aware of the existence of the other applications ([Column 2, Lines 33-36] of Harper).

Regarding **claim 27**, Rawat clearly shows and discloses a computer program product comprising instructions embodied on a memory of a computer system that cause at least one processor of the computer system to execute a method ([Column 11, Lines 59-64]), the method comprising:

loading an application program into memory (*Client-side program code examines electronic documents such as web pages and automatically fills in fields of forms contained in the document with the appropriate data from a user profile, without requiring prior mapping or examination of the form*, [Column 3, Lines 61-65]. *It is well inherent that any computer program is loaded into computer's memory in advance to execution*), the application program comprising a variable that is defined as an instance of both a basic data type and a specific data type (*Figure 1 shows the 'Credit Card*

Number' field. This field is a variable that is an instance of both basic data type, i.e. integer, and specific data type, i.e. credit card number);

accessing metadata at runtime to map the variable to a definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 18-22]); and

dynamically adding the basic data type of the variable with metadata at runtime (*filling out electronic forms automatically, in which the fields of an HTML form are identified and mapped to the correct user data based on visible form elements such as field labels. Following mapping and identification, the fields of the form are populated with the correct user data ,without reference to a previous, stored mapping or analysis of the form, and without requiring user intervention*, [Column 4, Lines 56-62]); and

processing the variable consistently with the metadata definition of the specific data type (*the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301. Also incorporated in the field label dictionary are the rules for mapping the field to the correct metadata and for mapping fields lacking labels to the correct metadata based on the field's context. At the same time, the functional blocks of the form are identified, for example, shipping address and*

billing address. In this way, identical field labels, such as "address" or "zip code" in a shipping address block and a billing address block are mapped to the correct metadata, [Colum 5, Lines 18-32]).

Rawat does not explicitly disclose the application program automatically processes changes in the metadata without incurring changes in a computer system implementing the method.

Harper discloses automatically processes changes in the metadata without incurring changes in a computer system (*Addition of messages or deletion of existing messages is controlled by the database 110, and the applications 102 which use the messages. Neither the MOM modules 104, nor the database 110, needs to be recompiled to add data entities in the database 110 such as Hosts, Applications, Queues, Queue Managers, etc. Messages and MetaData can also be added, modified, or deleted without redesign or recompilation of a system in accordance with one embodiment of the present invention. Further, in one embodiment, these changes can be made while the system is in use, without affecting the operation of other, unchanged parts of the system*, [Column 7, Lines 16-27]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Harper with the teachings of Rawat for the purpose of integrating multiple applications, where the various applications do not need to be aware of the existence of the other applications ([Column 2, Lines 33-36] of Harper).

Regarding **claim 28**, Rawat further discloses the metadata is stored on a hard disk, and the application program loaded into memory accesses the metadata by retrieving the metadata from the hard disk (*Logic 303 stored and executed on the client implements a probabilistic, rule-based method of analyzing the form in separate steps. In a first step, the logic traverses the form from beginning to end, locating the field labels, associating them with a field, and then mapping the field to the correct metadata based on a best match from a field label dictionary 304--a file of analogs, or expressions resembling the field label, stored on the client 301*, [Column 5, Lines 15-23]).

7. **Claims 4**, and **15**, are rejected under 35 U.S.C. 103(a) as being unpatentable over Rawat et al. (*Pat. No. 6,662,340, filed on May 30, 2002; hereinafter Rawat*) in view of Harper et al. (*Pat. No. US 7,093,261, filed on July 25, 2001; hereinafter Harper*), and further in view of Koseki et al. (*Pat. No. US 6,732,124, filed on February 9, 2000; hereinafter Koseki*).

Regarding **claim 4**, and **15** Rawat, as modified by Harper, does not explicitly disclose a metadata service copies the metadata to a metadata cache.

However, Koseki discloses a metadata service copies the metadata to a metadata cache (*a metadata cache is provided as part of the computer's main memory to hold a copy of metadata objects from the metadata volume. A metadata loading unit reads out a specific metadata object from the metadata volume to the metadata cache when a transaction demands it*, [Column 9, Lines 48-58]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Koseki with the teachings of Rawat, as modified by Harper, for the purpose of preventing unwanted changes in the original metadata by modifying the copy of metadata in the metadata cache instead of directly manipulating the original in the metadata volume ([Column 9, Lines 55-58] of Koseki).

8. **Claims 9-10, 20-21, 31-33,** and **36-38** are rejected under 35 U.S.C. 103(a) as being unpatentable over Rawat et al. (*Pat. No. 6,662,340, filed on May 30, 2002; hereinafter Rawat*) in view of Harper et al. (*Pat. No. US 7,093,261, filed on July 25, 2001; hereinafter Harper*), and further in view of Logan et al. (*Pub. No. US 2003/0093790, filed on June 8, 2002; hereinafter Logan*).

Regarding **claims 9**, and **20**, Rawat, as modified by Harper, does not explicitly disclose the metadata is stored along with the application program in a private instance of the metadata.

However, Logan discloses the metadata is stored along with the application program in private instance of the metadata (*metadata contributed by other users and stored in a public database as well as private database. The metadata stored may be created, edited and deleted using a Web server or other server operates to contribute to the metadata*, [0308]). *The playback control 211 of the Personal Video Recorder (PVR) controls the playback of stored video programming seen at 217, stored electronic program guide (EPG) data seen at 218, application data such as standard templates stored at 220, metadata describing programs and program segments stored at 221, and other system control data stored at 222*, [0301]-[0304]).

Logan further discloses a user can perform actions leading to changes in the metadata without impacting other users of the application (*The metadata stored at 251-255 may be created, edited and deleted using a Web server 261 or other server 262 operated by the metadata service to provide the ability for employees of the metadata service to create and modify the stored metadata as indicated at 265*, [0308]. *It is clear that since the public does not have access to the private instance of metadata, changes to the metadata made by a private user will not affect users of the public*).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Logan with the teachings of Rawat, as modified by Harper, for the purpose of distributing, recording, organizing and editing metadata that is used to selectively distribute, record, organize, edit and play program content ([0004] of Logan).

Regarding **claims 10**, and **21**, Rawat, as modified by Harper, does not explicitly disclose storing metadata in public instance of the metadata.

However, Logan discloses the metadata is stored along with the application program in a public instance of the metadata (*metadata contributed by other users and stored in a public database as well as private database. The metadata stored may be created, edited and deleted using a Web server or other server operates to contribute to the metadata*, [0308]). *The playback control 211 of the Personal Video Recorder (PVR) controls the playback of stored video programming seen at 217, stored electronic program guide (EPG) data seen at 218, application data such as standard templates*

stored at 220, metadata describing programs and program segments stored at 221, and other system control data stored at 222, [0301]-[0304]).

Logan further discloses a user can introduce changes in the metadata that affect all users of the application program (*The metadata stored at 251-255 may be created, edited and deleted using a Web server 261 or other server 262 operated by the metadata service to permit the public (other viewers) to contribute to the metadata as illustrated at 264*, [0308]. *It is clear that since a public member can modify the metadata stored in a public instance, changes made by one public user will affect all public users*).

Regarding **claims 31**, and **36**, Logan further discloses the metadata is stored together with other metadata in a metadata store, and the metadata store is unknown to an application developer at design time (*Metadata created by users may be shared directly between users. When shareable metadata exists at a user location, it may be "registered" by supplying its resource address (such as an Internet URL) to the remote location which then relays the URL to other users who directly access the descriptive metadata from the other user's metadata storage 133 in a peer-to-peer transfer. In this form, the remote facility shown in FIG. 1 operates as a registry or directory that permits users to share descriptive metadata about broadcast programming with one another on a community basis*, [0092].

Regarding **claims 32**, and **37**, Harper further discloses providing metadata to the application program before and after changing the metadata store at runtime of the application program without causing a system failure (*Addition of messages or deletion of existing messages is controlled by the database 110, and the applications 102 which*

use the messages. Neither the MOM modules 104, nor the database 110, needs to be recompiled to add data entities in the database 110 such as Hosts, Applications, Queues, Queue Managers, etc. Messages and MetaData can also be added, modified, or deleted without redesign or recompilation of a system in accordance with one embodiment of the present invention. Further, in one embodiment, these changes can be made while the system is in use, without affecting the operation of other, unchanged parts of the system, [Column 7, Lines 16-27]).

Regarding **claims 33**, and **38**, Logan further discloses changing a second implementation portion of metadata services on a side of the metadata store without changing a first implementation portion of metadata services on a side of an integrated development environment (*The metadata stored at 251-255 may be created, edited and deleted using a Web server 261 or other server 262 operated by the metadata service to provide the ability for employees of the metadata service to create and modify the stored metadata as indicated at 265*, [0308]. *It is clear that since the public does not have access to the private instance of metadata, changes to the private metadata storage made by a private user will not affect the public metadata storage*).

9. **Claims 29-30**, and **34-35** are rejected under 35 U.S.C. 103(a) as being unpatentable over Rawat et al. (*Pat. No. 6,662,340, filed on May 30, 2002; hereinafter Rawat*) in view of Harper et al. (*Pat. No. US 7,093,261, filed on July 25, 2001; hereinafter Harper*), and further in view of Mojsilovic et al. (*Pub. No. US 2003/0195883, filed on April 15, 2002; hereinafter Mojsilovic*).

Regarding **claims 29**, and **34**, Rawat, as modified by Harper, does not disclose the metadata is stored in a backend system such that the application program executes on the backend system and the user interface is assembled in a different system that is separate from the computer system.

However, Mojsilovic discloses the metadata is stored in a backend system such that the application program executes on the backend system (*server back-end 120B then accesses and retrieves the image using the returned URL, computes the image features (image metadata, as described above with respect to FIGS. 15A and 15B) and stores the image metadata in the repository 124 along with associated image URLs and preferably the URLs of image-related and referenced documents*, [0146]) and the user interface is assembled in a different system that is separate from the computer system (*Logical part 120C is the server front-end. The front-end of the search engine runs on the HTTP server and searches the repository according to the queries of the users. Users interact with the Internet portal 120 through a graphical user interface (GUI) 126, typical for Internet search engines*, [0119]-[0120]).

It would have been obvious to a person skilled in the art at the time of the invention to incorporate the teachings of Mojsilovic with the teachings of Rawat, as modified by Harper, for the purpose of automatically characterizing images according to their modalities, and that also employs semantic information for browsing, searching, querying and visualizing collections of digital images ([0010] of Mojsilovic).

Regarding **claims 30**, and **35**, Mojsilovic further discloses the different system is a portal system using a portal runtime framework (*Figure 10 is a high level view of an Internet portal that provides access to the database*, [0033]).

Conclusion

10. **THIS ACTION IS MADE FINAL**. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Contact Information

11. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Son T. Hoang whose telephone number is (571) 270-1752. The Examiner can normally be reached on Monday – Friday (7:00 AM – 4:00 PM).

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Christian Chace can be reached on (571) 272-4190. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Son T Hoang/
Examiner, Art Unit 2165
October 27, 2008

/S. P./
Primary Examiner, Art Unit 2164

/Christian P. Chace/
Supervisory Patent Examiner, Art Unit 2165